WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Crop Prediction with Soil Health Analysis: An Integrated Machine Learning Platform for Agro-Climatic Crop Recommendation, Soil Nutrient Balancing and Farm Economic Simulation

Yuganshu Sukare , Pranay Nimkar , Nishant Warthi

Department of Computer Science & Engineering, G H Raisoni University, Amravati, Maharashtra, India
yuganshu.sukare.cse@ghrua.edu.in , pranaynimkar2004@gmail.com , nishant.warthi.cse@ghrua.edu.in

ABSTRACT—

Selecting an appropriate crop and applying balanced fertilizer are persistent challenges for Indian farmers because soil fertility, temperature, humidity and rainfall vary widely across regions and seasons. Most crop recommendation tools return only a crop label and leave nutrient management and profitability unaddressed. This paper presents an integrated, web-based decision-support platform that combines (i) a Random Forest classifier for multiclass crop recommendation from seven inputs (nitrogen, phosphorus, potassium, soil pH, temperature, relative humidity and rainfall), (ii) a rule-based Soil Health Index (SHI) on a 0–100 scale with nutrient deficiency detection, (iii) a fertilizer module that converts deficiencies into estimated quantities of urea, diammonium phosphate (DAP) and muriate of potash (MOP), and (iv) an economic module that estimates revenue, net profit and return on investment (ROI). The classifier, a 100-tree Random Forest trained on 2,200 records covering 22 crop classes with a stratified 80:20 split, achieved a test accuracy of 99.30%, and stratified five-fold cross-validation was used to assess stability. The system is implemented in Python with Flask RESTful APIs and a JavaScript front end. Worked examples demonstrate crop recommendation with a confidence score, an SHI of 67/100 with nitrogen and potassium deficiencies, fertilizer quantities for a two-acre plot, and an illustrative profit estimate. We discuss the limits of the benchmark dataset, the advisory nature of rule-based outputs, and the validation required before field deployment.

***Keywords*—crop recommendation, machine learning, Random Forest, soil health index, precision agriculture, fertilizer recommendation, decision support system, smart farming**

1. INTRODUCTION

Agriculture underpins India's economy and food security, yet productivity depends on many interacting factors: soil fertility, climate, water availability and the choice of crop. Traditional decisions rely on inherited experience, which is valuable but increasingly strained by shifting weather patterns and heterogeneous soils. Unbalanced fertilizer use compounds the problem, raising input costs while degrading long-term soil health.

Machine learning (ML) can learn the relationship between soil-climate conditions and crop suitability from historical data and has become a core technique in precision agriculture [1]–[4]. However, many published crop recommendation systems stop at predicting a crop. A farmer must also know whether the soil is deficient in key nutrients, how much fertilizer to apply, and whether the crop is likely to be financially worthwhile. Providing these answers in a single tool lowers the barrier to data-driven decisions, particularly for smallholders who have limited access to agronomists and laboratory services.

This paper therefore presents a unified platform that couples an ML classifier with agronomic rules and an economic estimator.

The system is intended to support, not replace, laboratory soil testing and professional agronomic advice.

A. Problem Statement:

Farmers need reliable information on soil and climate before choosing crops and fertilizers. Conventional decision-making may not consistently account for all relevant parameters, and tools that output only a predicted crop do not address nutrient deficiencies, fertilizer requirements or economics. A system is needed that integrates crop prediction, soil health evaluation, nutrient deficiency analysis and basic economic estimation.

B. Objectives:

1. Develop an ML model that predicts suitable crops from soil and climatic parameters.
2. Implement and evaluate a Random Forest classifier for multiclass crop classification.

3. Assess soil health from N, P, K and pH.
4. Identify nutrient deficiencies and recommend fertilizers.
5. Estimate cultivation cost, revenue and ROI.
6. Deploy the system as a responsive, user-friendly web application.

C. Contributions:

The main contributions of this work are:

1. A Random Forest crop recommendation model over 22 crop classes using seven soil and weather features, with class-probability-based confidence and alternative crop suggestions.
2. A transparent, rule-based Soil Health Index and nutrient deficiency analysis benchmarked against ICAR reference guidance [16].
3. A fertilizer quantity estimator for urea, DAP and MOP, scaled to field area and expressed in 50 kg bags.
4. A farm economic simulator reporting revenue, net profit and ROI.
5. A responsive web implementation using Flask RESTful APIs, enabling independent maintenance of the model, diagnostic and economic components.

D. Organization of the Paper:

Section II reviews related work and identifies the research gap. Section III describes the system design and architecture. Section IV details the methodology, including the dataset, the Random Forest model, the Soil Health Index, fertilizer and economic modules. Section V covers implementation and testing. Section VI presents results and discusses limitations and threats to validity, and Section VII concludes with future work.

2. Literature Review

A. Machine Learning in Agriculture:

Agricultural ML commonly uses soil nutrient levels, temperature, rainfall, humidity and pH as predictors. Classification algorithms predict categorical outcomes such as the most suitable crop, whereas regression estimates quantities such as yield. A systematic review by van Klompenburg et al. [1] surveyed ML approaches to crop yield prediction and highlighted the importance of environmental and agricultural features, while broader surveys by Liakos et al. [2] and Chlingaryan et al. [3] cover ML applications across crop, soil and nutrient management. Big-data oriented reviews position such analytics within the wider smart-farming ecosystem [4].

Decision trees, Random Forest [5], Support Vector Machines (SVM), k-Nearest Neighbors (k-NN) and gradient boosting [6],

[10] are frequently applied to structured agricultural data, and tree-based ensembles often remain strong on tabular problems [9]. Performance nevertheless depends on the dataset, feature selection, preprocessing and evaluation protocol, which is why this work reports a held-out test set together with cross-validation [11].

B. Crop Recommendation and Soil Diagnosis Studies:

Ensemble and tree-based models are widely used for crop recommendation and soil-based diagnosis because they handle nonlinear relationships between nutrients and climate variables without extensive feature engineering. Precision-agriculture work has additionally examined nitrogen status estimation, nutrient management and soil health assessment [3], [16]. Most of these studies treat crop prediction as the end point: the model returns a label, and the translation of that label into fertilizer plans and expected returns is left to the user. Soil health indices, in turn, are usually studied as stand-alone agronomic tools rather than as components of an automated recommender.

C. Existing Approaches and Their Limitations:

Table I summarizes the capabilities of the three groups of tools farmers typically encounter and compares them with the proposed platform. The comparison is qualitative and based on the functions each group is designed to provide.

TABLE I. Capability comparison of existing approaches and the proposed platform

Capability	**Crop-label recommenders**	**Yield predictors**	**Lab soil-test reports**	**Proposed platform**
Crop recommendation	Yes	No	No	Yes
Confidence and alternatives	Rarely	No	No	Yes
Soil health score	No	No	Partial	Yes (SHI 0–100)

Capability	Crop-label recommenders	Yield predictors	Lab soil-test reports	Proposed platform
Nutrient deficiency flags	No	No	Yes	Yes
Fertilizer quantity (bags)	No	No	Advisory	Yes
Revenue, profit and ROI	No	Partial	No	Yes
Single web interface	Varies	Varies	No	Yes

D. Research Gap:

Existing recommenders mainly output a crop name. Crop selection is only one component of farm planning; farmers also need nutrient deficiency information, fertilizer quantities, cultivation costs and expected returns. The proposed platform addresses this integration gap by combining ML-based crop classification with rule-based soil health assessment, fertilizer planning and economic estimation within a single web-based system.

3. System Design

The platform follows a modular, layered architecture, shown in Fig. 1. Separating the interface from the backend allows model inference and agronomic calculations to be updated independently, for example refreshing market prices without retraining the model. The design has four logical layers: the user interface, the application layer, the analytical modules and the data/configuration store.

A. Frontend:

The frontend is the presentation layer. It is built using HTML5, CSS3 and JavaScript. HTML5 provides the structure of the input form and the result views; CSS3 controls the visual design and ensures that the platform works smoothly on desktops, tablets and mobile phones; JavaScript validates entries on the client side, calls the backend asynchronously so that pages update without a full reload, and uses Chart.js [20] to draw the soil health and economic charts.

B. Backend (Application Layer):

The backend is the logic layer. It is written in Python and exposes Flask RESTful APIs [15], [19]. Its responsibilities are input validation, preprocessing, invoking the trained model, coordinating the soil, fertilizer and economic modules, and returning results as JSON. Because every module is reached through its own endpoint, a module can be revised (for example, new fertilizer prices) without touching the others.

C. Machine Learning Module:

The ML module loads the trained 100-tree Random Forest at start-up and keeps it in memory. For each request it returns the predicted crop together with class probabilities, from which a confidence score and the top-k alternative crops are derived.

D. Soil Health, Fertilizer and Economic Modules:

The soil health diagnostic compares N, P, K and pH with configured reference ranges and computes the SHI. The fertilizer module converts deficiencies into quantities of urea, DAP and MOP. The economic module uses the recommended crop, the area and the fertilizer quantities, together with configurable yield, price and cost assumptions, to estimate revenue, net profit and ROI.

E. Data / Configuration Store:

The data store holds the trained model file and a configuration set containing nutrient thresholds, fertilizer composition, and yield, price and cost assumptions. Keeping these values outside the code is what allows agronomic or market changes to be applied without retraining.

F. Working of All the Layers Together:

- The user interacts with the frontend built using HTML5, CSS3 and JavaScript.
- Actions such as submitting the soil and weather form send a request to the backend, handled by Python and Flask.
- The backend validates and preprocesses the inputs and passes the feature vector to the Random Forest.
- The soil, fertilizer and economic modules read thresholds and assumptions from the configuration store.
- The backend assembles the results as JSON and returns them.
- The frontend displays the crop, alternatives, SHI, fertilizer plan and economic outlook.

G. Architecture Design:

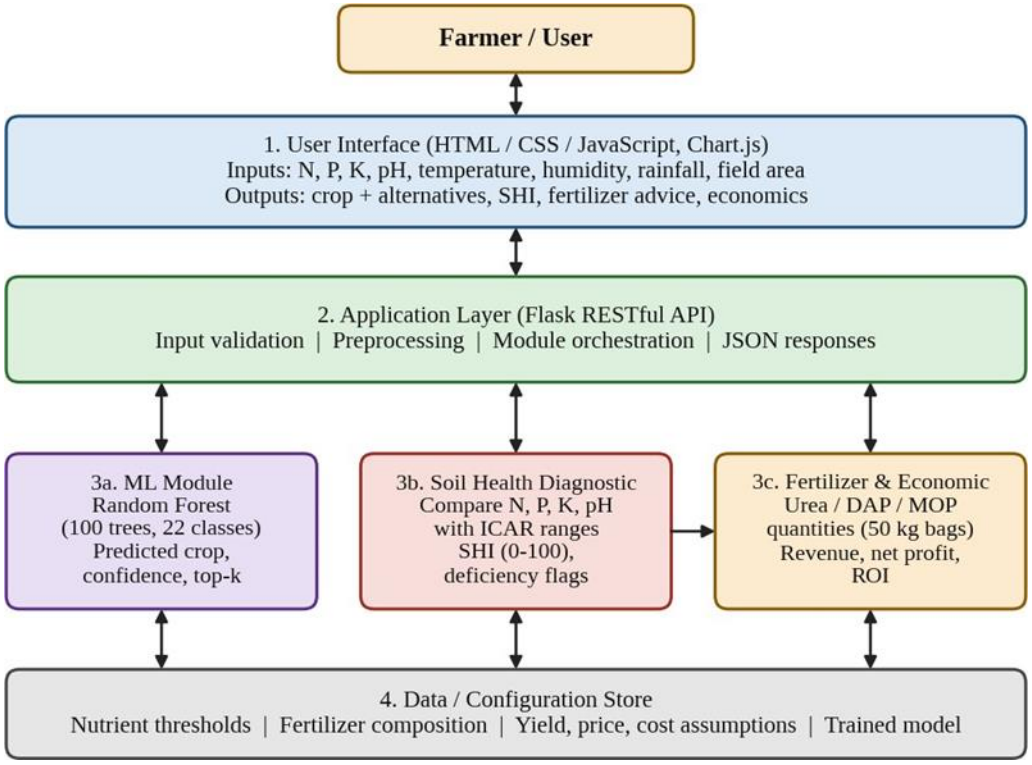


Fig. 1: Layered architecture of the proposed platform. The Flask REST API validates inputs and coordinates the Random Forest, Soil Health Diagnostic and Fertilizer/Economic modules, which read thresholds and assumptions from a shared configuration store.

TABLE II. Layer responsibilities

Layer	**Responsibility**
User Interface	Collects N, P, K, pH, temperature, humidity, rainfall and field area; displays the recommended crop, alternatives, soil health, fertilizer advice and economics
Application (Flask REST API)	Validates inputs, invokes the model, coordinates the soil and economic modules, returns JSON to the interface
Machine Learning	Trained Random Forest; outputs the predicted class and class probabilities
Soil Health Diagnostic	Compares N, P, K, pH with reference ranges; computes the SHI (0–100); flags deficiencies
Fertilizer and Economic	Converts deficiencies to urea/DAP/MOP quantities; estimates revenue, profit and ROI
Data / Configuration	Stores thresholds, fertilizer composition, yield and price assumptions, and the trained model

H. Data Flow:

Fig. 2 shows the data flow at process level. User inputs enter through process P1 (validation and preprocessing). The validated vector is sent to the classifier (P2) and the soil health process (P3). Deficiency flags from P3 drive the fertilizer estimator (P4), whose bag quantities, together with the recommended crop from P2, feed the economic simulator (P5). Process P6 packages all outputs as JSON for the interface. The trained model (D1) and the configuration store (D2) are read-only during a request.

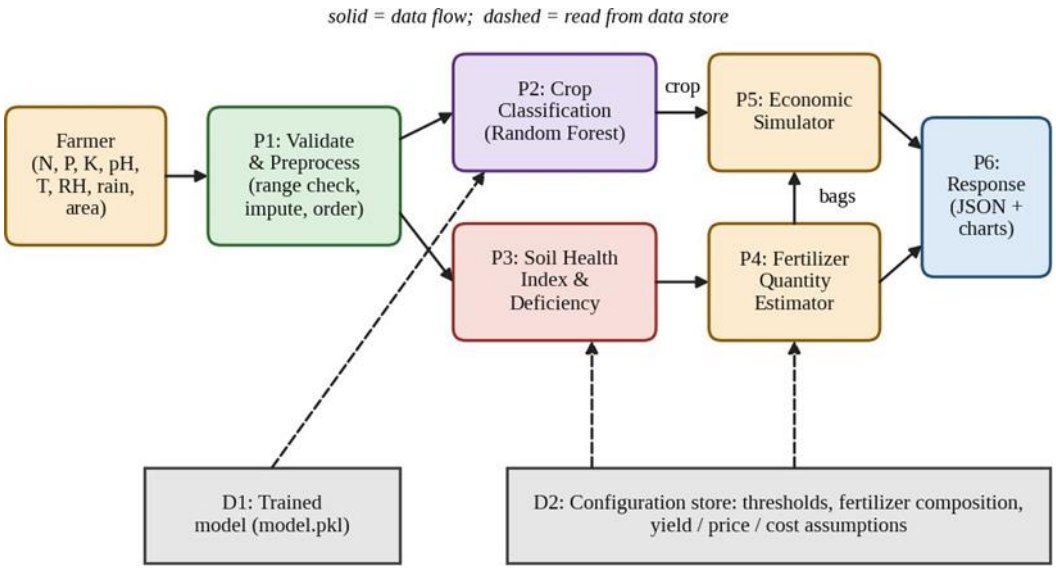


Fig. 2: Data flow diagram of the platform (processes P1–P6, data stores D1–D2).

I. Use-Case Model:

Two actors interact with the platform (Fig. 3). The farmer or advisor enters inputs and consumes the recommendation, soil health, fertilizer and economic views. The administrator maintains the configuration (thresholds, prices) and replaces or retrains the model; these functions are kept separate from farmer-facing functions.

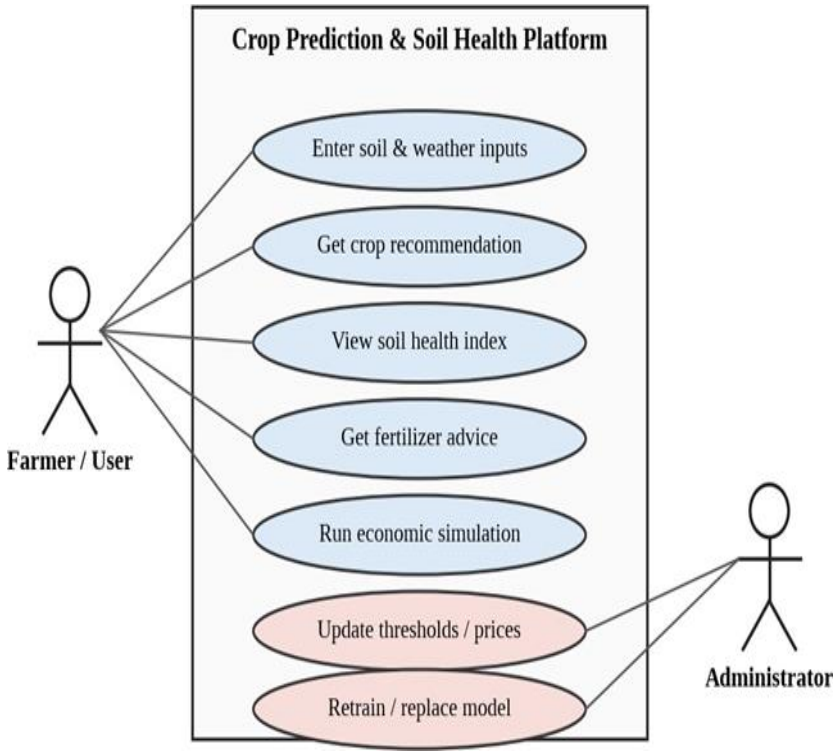


Fig. 3: Use-case diagram showing farmer and administrator interactions.

J. Request Sequence:

A single analysis request follows the sequence in Fig. 4. The front end posts the form values once; the Flask API orchestrates all downstream calls, so the client does not need to know about individual modules.

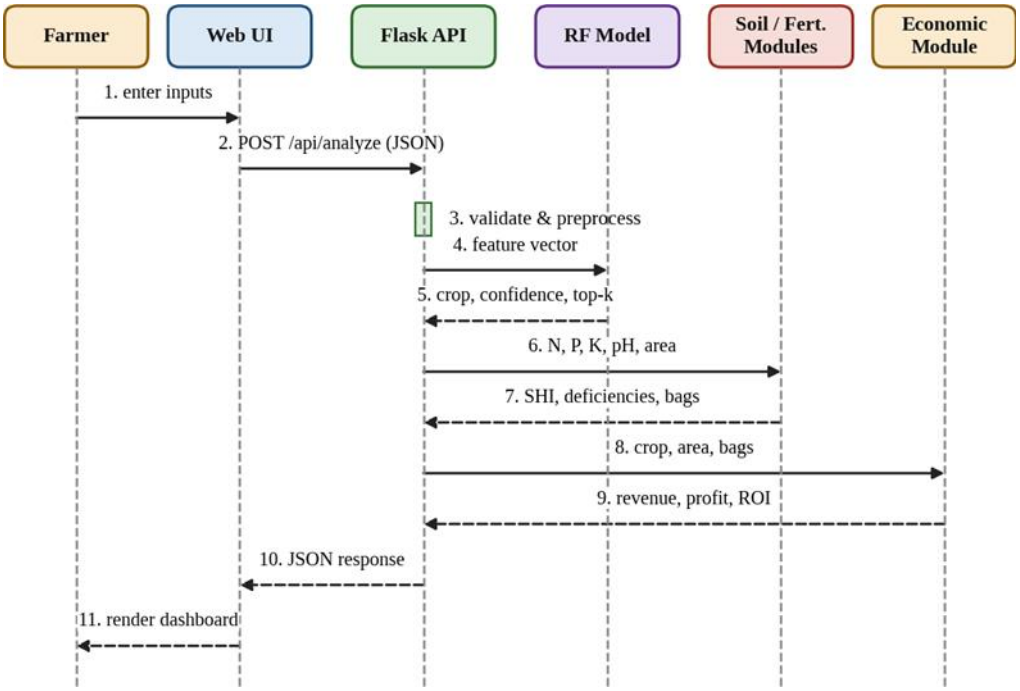


Fig. 4: Sequence diagram of one analysis request.

K. Design Considerations:

Four principles guided the design. *Modularity:* every analytical step is a separate module behind its own endpoint, so thresholds, fertilizer grades or market prices can change without retraining. *Statelessness:* each request carries only agronomic values (soil, weather and area), so the API stores no personal data and scales by simple replication. *Transparency:* the SHI, fertilizer and economic modules are rule-based and every number can be traced to a configured threshold or assumption, which matters for advice that influences spending. *Accessibility:* the interface is responsive and uses plain labels so that it can be operated from a low-end mobile phone.

4. Methodology

The overall methodology is summarized in Fig. 5. The dataset is validated and preprocessed, split with stratification, used to train a Random Forest, assessed with cross-validation and a held-out test set, and finally deployed behind the Flask API.

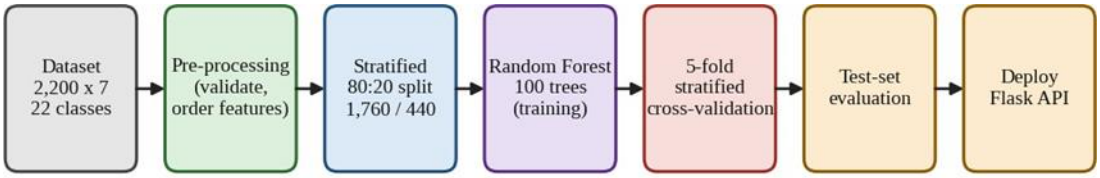


Fig. 5: Methodology pipeline from dataset to deployment.

A. Dataset:

The dataset contains 2,200 records across 22 crop classes (100 records per class when balanced), each with seven features, and the target is the crop class. It follows the structure of the public Crop Recommendation dataset [17]. The 22 classes are rice, maize, chickpea, kidney beans, pigeon peas, moth beans, mung bean, black gram, lentil, pomegranate, banana, mango, grapes, watermelon, muskmelon, apple, orange, papaya, coconut, cotton, jute and coffee. Fig. 6 shows the balanced class distribution; Table III lists the features and the value ranges reported for the public version of the dataset.

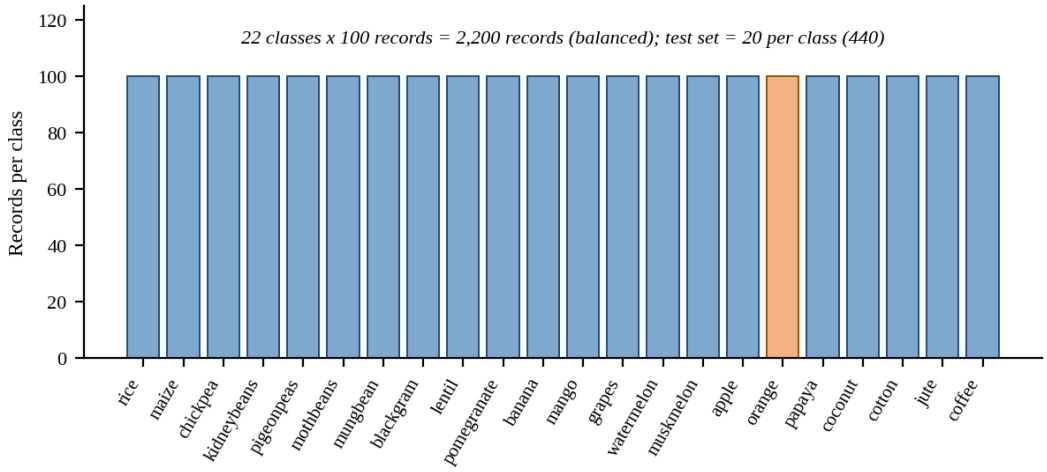


Fig. 6: Class distribution of the dataset (100 records per class).

TABLE III. Input features and summary statistics of the public dataset [17]

Feature	Description	Min	Mean	Max
N	Soil nitrogen content	0	50.6	140
P	Soil phosphorus content	5	53.4	145
K	Soil potassium content	5	48.2	205
Temperature	Ambient temperature (°C)	8.8	25.6	43.7
Humidity	Relative humidity (%)	14.3	71.5	100.0
pH	Soil acidity / alkalinity	3.5	6.5	9.9
Rainfall	Rainfall (mm)	20.2	103.5	298.6

Dataset quality and diversity directly influence generalization to new agricultural conditions. The dataset was checked for duplicates, inconsistent units and potential target leakage.

B. Preprocessing:

The preprocessing pipeline performs the following steps at training time and again for every request:

1. Confirm that required fields are present.
2. Validate numeric type and acceptable parameter ranges.
3. Identify missing or invalid weather values.
4. Apply the configured regional weather imputation where appropriate (weather data may be sourced from a service such as Open-Meteo [18]).
5. Order features identically to training.

Any imputation is fitted on training data only to avoid information leakage.

C. Random Forest Classifier:

Random Forest [5] is an ensemble of decision trees, each trained on a bootstrap sample [7] and considering a random subset of features at each split. For classification, the trees vote and the majority class is returned (Fig. 7):

$$\hat{y} = \text{mode} \{ T_1(x), T_2(x), \ldots, T_n(x) \} \quad (1)$$

where x is the seven-element feature vector, $T_i(x)$ is the class predicted by tree i, n = 100 is the number of trees and $\hat{y}$ is the predicted crop. Class probability estimates, obtained from the trees' per-class probabilities (equal to the fraction of trees voting for a class when trees are grown to purity), provide the recommendation confidence and rank the alternative crops.

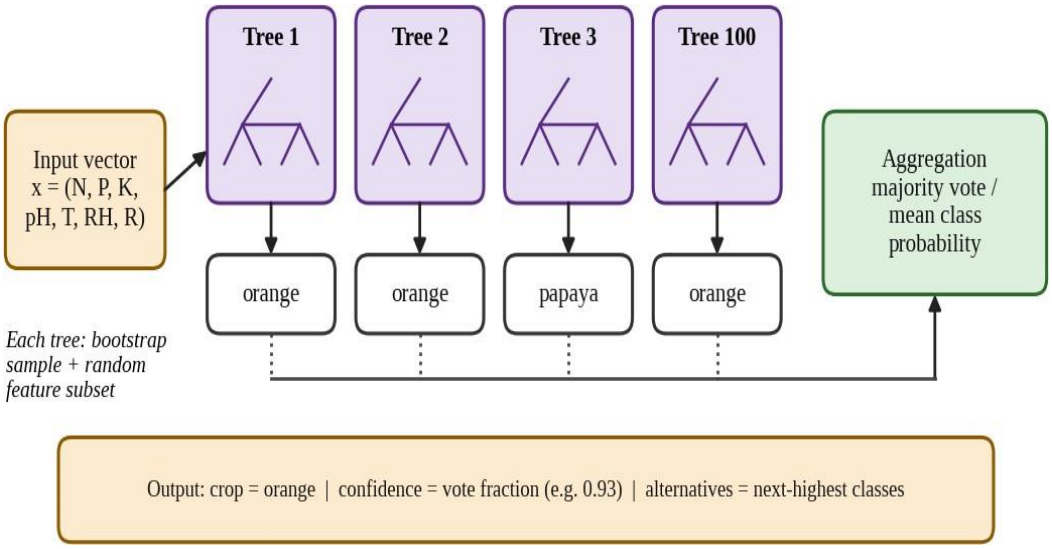


Fig. 7: Random Forest voting for one input vector (illustrative).

Random Forest suits this problem because it models nonlinear relationships and feature interactions, needs little scaling or feature engineering, and reduces the overfitting risk of a single tree [5], [14]. The model was implemented with scikit-learn [8]; hyperparameters other than the number of trees (100) were left at their library defaults.

D. Training and Validation:

The data were split with a stratified 80:20 train-test partition (1,760 training and 440 test records), preserving class proportions, so that each class contributes 20 records to the test set. Stability was examined with stratified five-fold cross-validation [11]. Preprocessing was fitted within training folds only, and the test set was held out from model selection and tuning. Accuracy is

Accuracy = (Correct Predictions / Total Predictions) × 100 (2)

Because accuracy can hide weak classes, per-class precision, recall and F1-score are also computed from the confusion matrix, where TP, FP and FN are the true positives, false positives and false negatives of a class:

Precision = TP / (TP + FP) , Recall = TP / (TP + FN)(3)

F1 = 2 × Precision × Recall / (Precision + Recall) (4)

The macro-average of these scores weights every crop equally, which suits a balanced 22-class problem. With 20 test records per class, a single error changes a class's recall by 5 percentage points, so any class with errors is examined individually in the confusion matrix.

E. Prediction Workflow:

1. The user enters soil and weather parameters.
2. The application validates the values.
3. The validated vector is passed to the trained model.
4. The model predicts the most suitable crop.
5. Class probabilities give a confidence score and alternative crops.
6. The interface displays the recommendation with soil, fertilizer and economic advice.

F. Soil Health Index:

The diagnostic module compares measured N, P, K and pH against configured reference ranges, in line with ICAR nutrient indexing guidance [16], and flags deficiencies or imbalances. Each nutrient is classified as Low, Medium or High and assigned a sub-score s (Low = 33, Medium = 67, High = 100); pH receives 100 inside the optimum window for most field crops (about 6.0–7.5) and lower scores outside it (Table IV). The Soil Health Index is the weighted mean of the four sub-scores:

SHI = wNsN + wPsP + wKsK + wpHspH , wN = wP = wK = wpH = 0.25 (5)

TABLE IV. Soil health scoring scheme used by the diagnostic module

Parameter	**Low (s = 33)**	**Medium (s = 67)**	**High / optimum (s = 100)**
Available N (kg/ha)	< 280	280–560	> 560
Available P (kg/ha)	< 10	10–25	> 25
Available K (kg/ha)	< 108	108–280	> 280
Soil pH	outside 5.5–8.0	5.5–6.0 or 7.5–8.0	6.0–7.5

TABLE V. Interpretation bands for the Soil Health Index

SHI range	**Band**	**Interpretation**
0–39	Poor	Multiple nutrients or pH are far from the optimum; corrective action is needed before sowing
40–69	Moderate	One or two nutrients are deficient; targeted fertilizer correction is recommended
70–100	Good	Nutrients and pH are adequate; maintenance fertilization is sufficient

The index is a simplified indicator whose validity depends on the thresholds, soil type and reference standards adopted. The resulting score is mapped to the bands in Table V. A nutrient scoring Low is reported as a deficiency and triggers the fertilizer module. As an example, with N and K rated Low and P and pH rated optimum, SHI = 0.25 (33 + 100 + 33 + 100) ≈ 67, which falls in the Moderate band.

G. Fertilizer Recommendation:

For each deficient nutrient, the module computes the shortfall relative to the target level and converts it into commercial fertilizer using nutrient content: urea for nitrogen, DAP for phosphorus (and some nitrogen) and MOP for potassium (Fig. 8). With the standard grades in Table VI, the product quantities for a field of area A are

$$\text{Urea (kg)} = A \times \Delta N / 0.46 \,,\ \text{DAP (kg)} = A \times \Delta P_2O_5 / 0.46 \,,\ \text{MOP (kg)} = A \times \Delta K_2O / 0.60 \quad (6)$$

where Δ denotes the per-unit-area shortfall, and the nitrogen supplied by DAP (18%) is credited against the urea requirement. Quantities are rounded to standard 50 kg bags. These are advisory estimates and should be validated against crop-specific requirements and local soil test results before field application.

TABLE VI. Fertilizer grades used for conversion

Fertilizer	**N (%)**	**P2O5 (%)**	**K2O (%)**
Urea	46	–	–
DAP (diammonium phosphate)	18	46	–
MOP (muriate of potash)	–	–	60

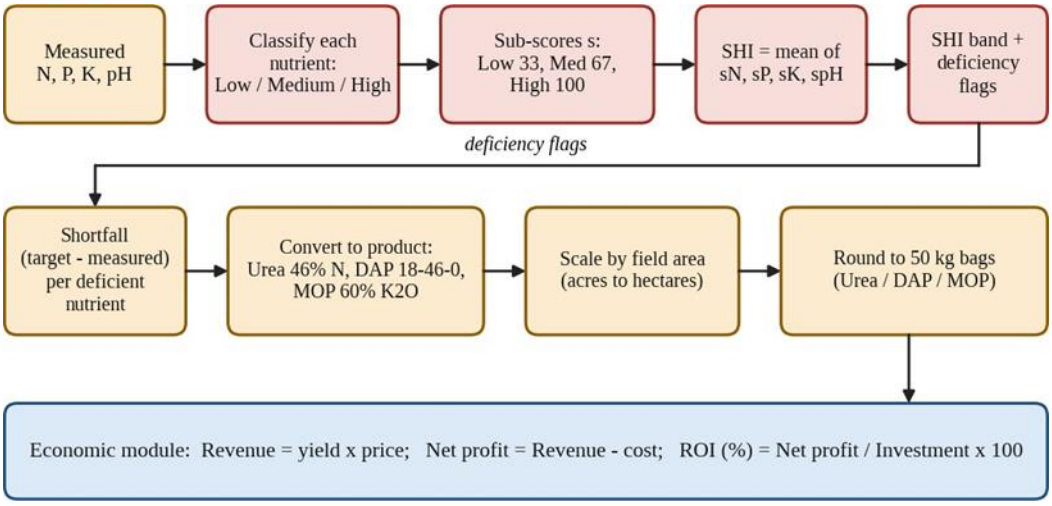


Fig. 8: Soil health scoring, deficiency detection, fertilizer conversion and economic estimation flow.

H. Economic Feasibility:

The economic module estimates revenue, net profit and ROI as

Revenue = Expected yield × Selling price per unit (7)

Net profit = Revenue − Total cultivation cost (8)

ROI (%) = (Net profit / Total investment) × 100 (9)

Yield, mandi prices and cultivation costs are configurable assumptions. Outcomes are preliminary planning estimates and will vary with weather, crop health, input costs and market prices.

5. Implementation

A. Tools and Technology:

The system is implemented in Python 3.12. The classifier is built with scikit-learn [8], Flask RESTful APIs [15], [19] expose prediction, soil diagnosis, fertilizer and economic endpoints, and a responsive JavaScript front end provides input forms and result presentation, with charts drawn by Chart.js [20]. Data handling uses pandas [13].

TABLE VII. Technologies used

Layer	**Technologies**
Frontend	HTML5, CSS3, JavaScript, Chart.js
Backend	Python 3.12, Flask RESTful API
Machine learning	scikit-learn (Random Forest), pandas
Weather data (optional)	Open-Meteo API for regional weather imputation
Configuration	Threshold, fertilizer-grade, yield, price and cost tables kept outside the code

B. REST API Design:

Each analytical function is available through its own JSON endpoint, and a combined endpoint orchestrates the full analysis in one call (Table VIII).

TABLE VIII. Representative REST endpoints

Method	**Endpoint**	**Purpose**
POST	/api/predict	Returns the recommended crop, confidence and top-k alternatives from the seven inputs
POST	/api/soil	Returns the SHI, per-nutrient class and deficiency flags

Method	Endpoint	Purpose
POST	/api/fertilizer	Returns urea, DAP and MOP quantities (kg and 50 kg bags) for a given area
POST	/api/economics	Returns revenue, net profit and ROI for a crop, area and cost assumptions
POST	/api/analyze	Runs all four steps and returns one combined JSON response

C. Key Features Implemented:

- Crop recommendation – predicted crop, confidence score and alternative crops from seven inputs.
- Soil health diagnostics – SHI on a 0–100 scale and nutrient deficiency flags.
- Fertilizer planning – urea, DAP and MOP quantities scaled to field area, in 50 kg bags.
- Economic analysis – revenue, net profit and ROI from configurable assumptions.
- Input validation – type, range and missing-value checks before any computation.
- Responsive interface – usable on desktop, tablet and mobile screens.

D. Work Implemented:

The web workflow comprises agricultural input forms, a crop recommendation view, soil health diagnostics, fertilizer calculations and economic analysis. Fig. 9 shows the layout of the dashboard: the input panel on the left and the four result cards on the right.

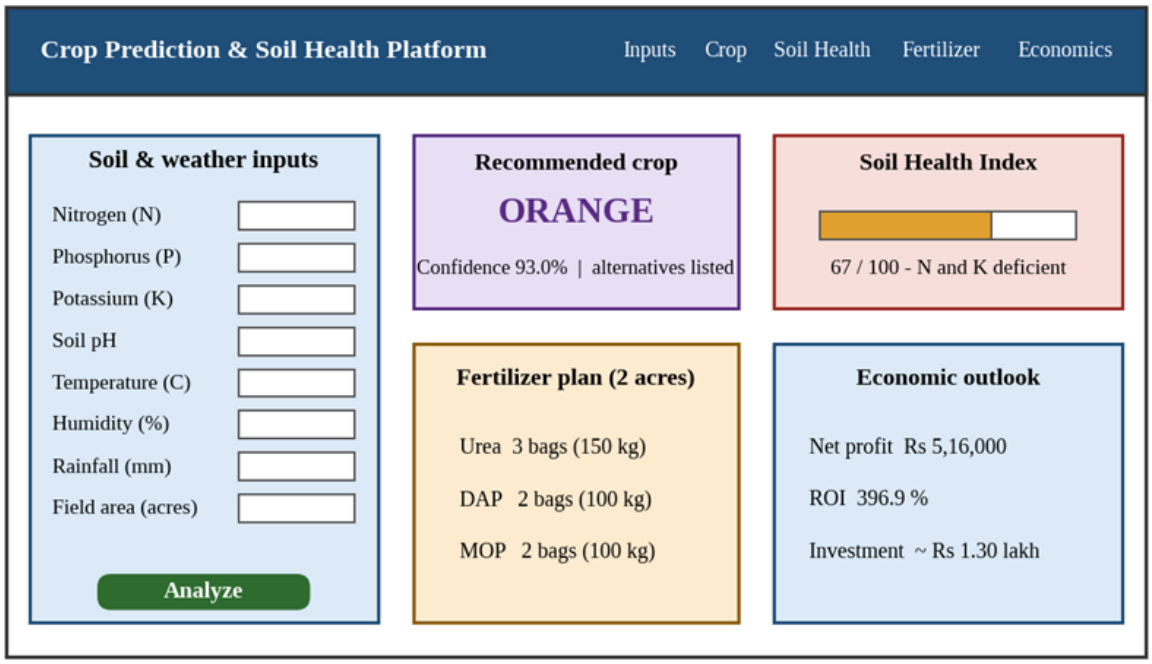


Fig. 9: Layout of the web dashboard showing inputs and result cards (schematic, using the worked example of Section VI).

E. Testing:

Testing covered input validation, prediction correctness, missing or invalid values, SHI and fertilizer calculations, API responses and interface behavior (Table IX). The system reports prediction inference in under 15 ms on the development machine.

TABLE IX. Test categories

Category	What was checked
Input validation	Numeric type, acceptable ranges and required fields are enforced; invalid entries are rejected with a message
Prediction correctness	Model output on known test records matches the offline evaluation
Missing / invalid values	Missing weather values are imputed or flagged rather than silently passed to the model

Category	What was checked
SHI and fertilizer calculations	Scores, deficiency flags and bag quantities match hand calculations
API responses	Each endpoint returns well-formed JSON and correct status codes
Interface behavior	Forms, charts and result cards render and update correctly on different screen sizes

6. Results and Discussion

A. Model Performance:

On the held-out test set the Random Forest achieved 99.30% accuracy across 22 crop classes (about 3 misclassifications out of 440 test records). Stratified five-fold cross-validation was used to examine stability, and the same protocol is applied to the baseline classifiers listed in Table X. Accuracy alone does not establish performance on every class or on conditions outside the dataset, so per-class precision, recall and F1-score and the confusion matrix should be inspected for the few confusable crops.

TABLE X. Classifier comparison (same split and protocol)

Model	Test accuracy (%)	5-fold CV mean ± SD (%)
Random Forest (100 trees)	99.30	–
Decision Tree	–	–
k-NN	–	–
SVM	–	–
Gradient Boosting / XGBoost	–	–

B. Crop Recommendation Example:

In a demonstration case the system recommended orange as the primary crop with a decision confidence of 93.0%, along with alternative crops. This illustrates how predictions and confidence are communicated; it is not evidence that orange suits every field with similar inputs. Because Random Forest vote fractions are not guaranteed to be calibrated probabilities [12], confidence should be interpreted cautiously.

C. Soil Health and Fertilizer Example:

For an example input the system produced an SHI of 67/100 and identified nitrogen and potassium deficiencies. For a two-acre field it recommended three 50 kg bags of urea (150 kg), two bags of DAP (100 kg) and two bags of MOP (100 kg). Using the grades in Table VI, these quantities supply the nutrient amounts in Table XI; DAP is applied as a basal dose that contributes phosphorus together with part of the nitrogen.

TABLE XI. Fertilizer recommendation and nutrients supplied (two-acre example)

Product	Quantity (kg)	Bags (50 kg)	Nutrient supplied
Urea	150	3	69 kg N
DAP	100	2	18 kg N and 46 kg P_2O_5
MOP	100	2	60 kg K_2O
Total	350	7	87 kg N, 46 kg P_2O_5, 60 kg K_2O

The agronomic accuracy of such outputs depends on the reference values, fertilizer composition and formulas configured, and on crop-specific requirements.

D. Economic Example:

Using projected yield, mandi selling prices and cultivation costs, an illustrative scenario produced an estimated net profit of

₹5,16,000 and an ROI of 396.9%, which implies a total investment of roughly ₹1.30 lakh and revenue of about ₹6.46 lakh (Fig. 10). These are scenario-specific estimates, not guaranteed profits.

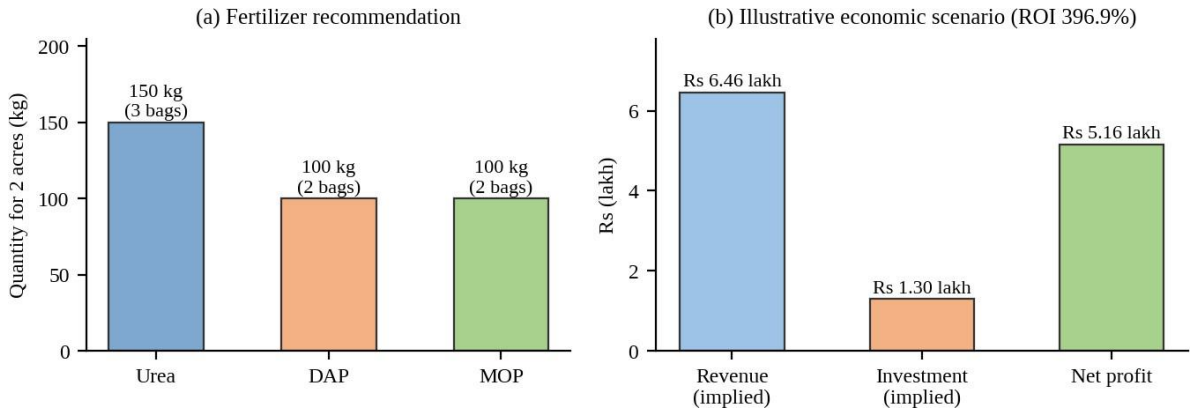


Fig. 10: (a) Fertilizer quantities and (b) illustrative economic outcome for the two-acre example.

E. Limitations and Threats to Validity:

- **Benchmark dataset.** The data are relatively small, class-balanced and well separated, so near-perfect accuracy is common on them and may not transfer to noisy field data from other regions and seasons.
- **Unit mapping.** The N, P, K values in the training data are on the dataset's own scale, whereas ICAR ratings are expressed in kg/ha; the mapping between them is a configurable assumption that needs calibration against laboratory tests.
- **Rule-based outputs.** The SHI, fertilizer and economic modules are advisory, depend on configured thresholds and prices, and do not replace soil testing or agronomist advice.
- **Confidence.** Vote fractions are not calibrated probabilities and should be read as relative support for a crop.
- **No field trial.** The platform has been evaluated on benchmark data and worked examples, not on a multi-season field study.

F. Discussion:

The platform's main strength is integration: rather than a crop label alone, users receive soil diagnostics, fertilizer planning and an economic outlook. The modular design means that thresholds, fertilizer grades and market assumptions can be updated independently of the trained model. The very high accuracy should, however, be interpreted with care. Independent validation on locally collected data is therefore essential before the recommendations are used for real purchasing decisions.

Compared with crop-label recommenders, the additional modules add value mainly at the decision stage: the deficiency flags tell a farmer what is wrong with the soil, the bag quantities translate that into a purchase list, and the economic outlook allows two candidate crops to be compared on expected return rather than suitability alone. Compared with laboratory soil-test reports, the platform is faster and cheaper but less precise, and it is best used as a first-level screening tool that guides which fields deserve a full laboratory test.

G. Summary of Reported Results:

TABLE XII. Summary of the key outcomes reported in this work

Item	**Reported outcome**
Dataset and split	2,200 records, 22 classes, 7 features; stratified 80:20 (1,760 train / 440 test)
Classifier	Random Forest, 100 trees; test accuracy 99.30%
Demonstration crop	Orange, decision confidence 93.0%, with alternative crops listed
Soil health example	SHI 67/100; nitrogen and potassium deficiencies flagged
Fertilizer example (2 acres)	3 bags urea (150 kg), 2 bags DAP (100 kg), 2 bags MOP (100 kg)
Economic example	Net profit ₹5,16,000; ROI 396.9% (implied investment about ₹1.30 lakh)
Inference time	Under 15 ms per prediction on the development machine

H. Usage Scenario:

A typical session proceeds as follows. A farmer or extension worker opens the dashboard on a phone, enters the soil test values for N, P, K and pH together with the local temperature, humidity and rainfall, and gives the plot area in acres. Within a fraction of a second the dashboard shows the

recommended crop with its confidence and two or three alternatives; the soil card shows the SHI and which nutrients are low; the fertilizer card lists the number of 50 kg bags to buy; and the economic card shows the expected net profit and ROI. If the farmer prefers the second-ranked crop, the economic card can be re-run for it, so the choice is made on suitability, soil condition and expected return together rather than on suitability alone.

7. Conclusion and Future Work

This paper presented an integrated web-based platform that goes beyond crop prediction. A 100-tree Random Forest recommends one of 22 crops from seven soil and weather inputs with a test accuracy of 99.30%; a rule-based Soil Health Index detects nutrient deficiencies; a fertilizer module converts them into urea, DAP and MOP quantities in 50 kg bags; and an economic module estimates revenue, net profit and ROI. The Flask-based architecture keeps these components independent and easy to maintain.

Future work will (i) complete and report per-class metrics and baseline comparisons, (ii) calibrate the confidence scores, (iii) replace configured thresholds with laboratory-validated, region-specific values, (iv) integrate live weather and mandi price feeds,

(v) add multilingual and voice-based access for farmers, and (vi) conduct multi-season field trials with extension officers before wider deployment.

ACKNOWLEDGEMENT

The authors would like to thank the Department of Computer Science & Engineering, G H Raisoni University, Amravati, for providing guidance, tools and infrastructure for this work.

REFERENCES

JOURNALS

[1]T. van Klompenburg, A. Kassahun, and C. Catal, “Crop yield prediction using machine learning: A systematic literature review,” *Comput. Electron. Agric.*, vol. 177, Art. no. 105709, 2020.

[2] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, “Machine learning in agriculture: A review,” *Sensors*, vol. 18, no. 8, Art. no. 2674, 2018.

[3] A. Chlingaryan, S. Sukkarieh, and B. Whelan, “Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review,” *Comput. Electron. Agric.*, vol. 151, pp. 61–69, 2018.

[4] A. Kamilaris, A. Kartakoullis, and F. X. Prenafeta-Boldú, “A review on the practice of big data analysis in agriculture,” *Comput. Electron. Agric.*, vol. 143, pp. 23–37, 2017.

[5] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[6] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[7] B. Efron, “Bootstrap methods: Another look at the jackknife,” *Ann. Statist.*, vol. 7, no. 1, pp. 1–26, 1979.

[8] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

CONFERENCE PROCEEDINGS

[9]L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on typical tabular data?” in *Proc. NeurIPS Datasets and Benchmarks Track*, 2022.

[10]T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[11]R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Proc. 14th Int. Joint Conf. Artif. Intell. (IJCAI)*, 1995, pp. 1137–1143.

[12]A. Niculescu-Mizil and R. Caruana, “Predicting good probabilities with supervised learning,” in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 625–632.

[13]W. McKinney, “Data structures for statistical computing in Python,” in *Proc. 9th Python in Science Conf. (SciPy)*, 2010, pp. 56–61.

TEXT BOOKS

[14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, NY, USA: Springer, 2009.

[15] M. Grinberg, *Flask Web Development*, 2nd ed. Sebastopol, CA, USA: O’Reilly Media, 2018.

REPORTS AND ONLINE RESOURCES

[16] Indian Council of Agricultural Research (ICAR), *Handbook of Agriculture*. New Delhi, India: ICAR.

[17] A. Ingle, "Crop Recommendation Dataset," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset

[18] Open-Meteo, "Open-Meteo free weather API." [Online]. Available: https://open-meteo.com/

[19] Pallets Projects, "Flask documentation." [Online]. Available: https://flask.palletsprojects.com/

[20] Chart.js Contributors, "Chart.js documentation." [Online]. Available: https://www.chartjs.org/docs/